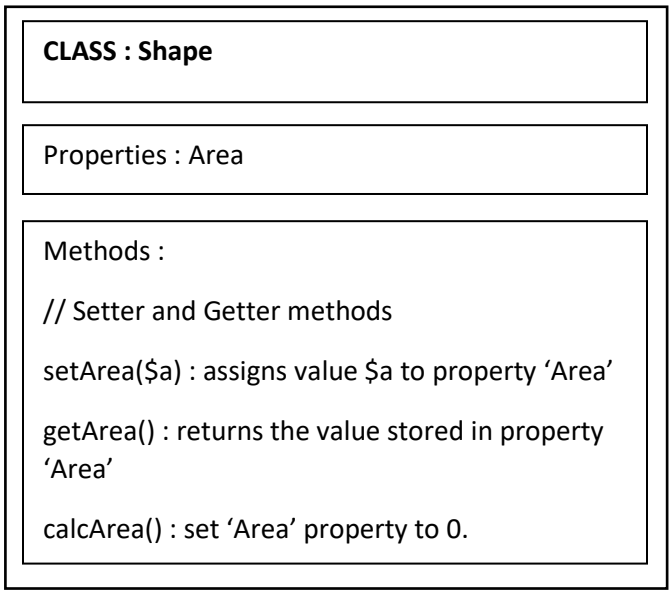
IDG2003 – Lab4

Exercise 1 : - This lab will focus entirely on Object Oriented Programming in PHP and has only 1 (long) exercise.

Part 1 : Creating the superclass/parent class

(a) In 1 php script, create a Shape class. (name it "Shape.php" for eg).
(b) Shape class should have an area property, a function to calculate area and setter and getter methods for the area property.
(c) The area property of Shape should be accessible only to the object's methods itself and not to those of subclasses (or outside).
(d) Methods of the Shape class should be accessible only to the object's methods itself and those of subclasses. (Note that, given that the area property can only be accessed from inside the class, the setter and getter methods will allow the subclasses to access the set a value to the area property and retrieve that value from the area property.
    a. For example : the setter method : setArea(), should simply take 1 argument(which would be the area) and save it in the area property of the Shape class.
    b. Similarly, the getter method : getArea(), should retrieve the value stored in the area property and return it.
(e) The area calculation method of Shape class should always set area to 0.



Part 2 : Creating the subclasses

(a) In their own respective php scripts, create 2 subclasses that extends from the 'Shape' class, named:
   I. Circle (defined in "Circle.php")
   II. Rectangle (defined in "Rectangle.php")

   (Note that since you are using separate files for classes' definitions, do not forget to include/require the relevant scripts where needed to ensure that your program works as expected). I would advise to fully focus on the creation of 1 subclass at a time, and ensuring that it works, before moving to the other.

(b) Rectangle and Circle classes should override the calcArea() method in the superclass accordingly and have necessary properties.
   I. For example, the circle subclass should have the radius property and a constant for pi (you can use the value of 3.14 for pi).
   II. Similarly, the rectangle subclass should have the width and length properties.
   III. Properties of Rectangle and Circle classes should not be accessible to outside but should be modifiable with a public interface. Create setter and getter methods to allow the properties to be set/retrieved publicly.
   IV. calcArea() should be implemented as follows for each subclass
      i. For the circle, area = pi * radius^2.
      ii. For the rectangle, area = width * length. (I hope this is obvious enough)
      iii. Also, calcArea() should call the setArea() method in the parent class to store the newly calculated value for area in the 'area' property in Shape.

(c) Since the getArea() method in the Shape superclass can only be accessed from within of itself and its subclasses, create another getArea() method in each of the subclasses that allows the parent getArea() method to be called publicly.

CLASS : Rectangle	Properties : Width, Length

Methods :

setWidth($w) : assigns $w to property 'Width'

setLength($l) : assigns $l to property 'Length'

getWidth() and getLength() returns stored value in Width and Length respectively

calcArea() : calculates area, calls the setArea() method in Shape superclass to set the value for Area.

getArea() : calls the getArea() method from Shape to retrieve the stored value for Area.

CLASS : Circle	Properties : Radius, Constant Pi

Methods :

setRadius($r) : assigns value $r to property 'Radius'

getRadius() : returns stored value in 'Radius'

calcArea() : calculates area, calls the setArea() method in Shape superclass to set the value for Area.

getArea() : calls the getArea() method from Shape to retrieve the stored value for Area.

Part 3 : Adding constructors

(a) Create 2 static properties in the Shape class that keeps track of the number of objects created for each subclass : such as $num_circ and $num_rect
(b) Create a constructor for the Shape super class.
(c) The Circle and Rectangle subclasses should also have constructors can access the Shape superclass constructor.
(d) In the Shape constructor, create a variable to check the type of object.

    For example, $classType which contains either “Rectangle” or “Circle” based on the value returned by the in-built php function get_class(). (Hint: get_class($this) would return the name of the subclass being constructed).

(e) Add a simple if/elseif statement to check if $classType is either “Rectangle” or “Circle”, and increments the respective static property created in (a).

Part 4 : Bringing it all together

(a) In a 4th php script, include the scripts for your subclasses.
(b) Create instances of Circles and Rectangles
(c) Using the setter methods you created, pass in values for radius or width and length into your subclasses and calculate area.
(d) Using the getArea method, obtain the stored Area value and print it.
(e) Print the values for the static variables in Shape to keep track of how many subclasses are being created. (You are free to print the values in any format you like, an example below)

A possible result could look like this :

```
This object is a Rectangle and has an area of 20
This object is a Circle and has an area of 6.28
This object is a Circle and has an area of 9.42
We now have 1 rectangles and 2 circles!
```